Handling Drifts and Shifts in On-Line Data Streams with Evolving Fuzzy Systems

E. Lughofer [a] P. Angelov [b]

[a] *Department of Knowledge-based Mathematical Systems, Johannes Kepler University of Linz, A-4040 Linz, Austria*
(email: edwin.lughofer@jku.at)

[b] *Intelligent Systems Research Laboratory, Department of Communication Systems, InfoLab21, Lancaster University, LA1 4WA, United Kingdom*
(email: p.angelov@lancaster.ac.uk)

Abstract

In this paper, we present new approaches to handling *drift* and *shift* in on-line data streams with the help of evolving fuzzy systems (EFS), which are characterized by the fact that their structure (rule base and parameters) is not fixed and not pre-determined, but is extracted from data streams on-line and in an incremental manner. When dealing with so-called *drifts* and *shifts* in data streams, one needs to take into account 1) automatic detection of *drifts* and *shifts*, and 2) automatic reaction to the *drifts* and *shifts*. This is important to avoid interruptions in the learning process and downtrends in predictive accuracy. To address the first problem, we propose an approach based on the concept fuzzy rule *age*. The second problem is addressed by including gradual forgetting of 1.) antecedent parts and 2.) consequent parameters. The latter can be achieved by including a forgetting factor in the recursive local learning process of the parameters, whose value is automatically extracted based on the intensity of the *shift/drift*. For addressing the former problem, we introduce two alternative methods: one is based on the evolving density-based clustering (*eClustering*) used to form the antecedents in the *eTS* approach; the other is based on the automatic adaptation of the learning rate of the evolving vector quantization (*eVQ*) method used to form the antecedent in the *FLEXFIS* approach. The paper concludes with an empirical evaluation of the impact of the proposed approaches in (on-line) real-world data sets in which *drifts* and *shifts* occur.

Key words: drifts and shifts in data streams, evolving fuzzy systems, eTS, FLEXFIS, detection and reaction to drifts and shifts, age of a cluster/fuzzy rule, gradual forgetting

1 Introduction

1.1 Motivation and State of the Art

Nowadays, data-driven fuzzy systems have become popular in many industrial applications, as in contrast to expert-based fuzzy systems, they can be generated automatically from process data such as measurements, images, and signal streams. Furthermore, they are proven universal approximators [46] (i.e. they can model a given problem to any degree of accuracy — theoretically, at least) that allow some insights in the form of linguistically [15] and visually [38] [16] interpretable rules.

During the last decade, the field of 'evolving fuzzy systems' (EFS) has emerged as an important topic in fuzzy systems research [1], as they are capable of including new information on demand and on-the-fly into models without necessarily using *prior* knowledge. EFS learn permanently from their environment, hence they may even be considered as a step towards true computational intelligence [6]. Furthermore, they are applicable in fast on-line identification [5] and modeling processes, and can be used with huge data bases which cannot be loaded into the virtual memory all at once [31]. Often, off-line data available in advance is simply not sufficient to build reliable models with high predictive quality; this may also necessitate the application of EFS as in [37].

Various EFS approaches have been established during the last years. One of the most popular and pioneering approach among them is the *eTS* family, which comes with a regression [4] [5] and a classification variant [2] exploiting various fuzzy model architectures [8]. These were further advanced in [1] [9] [3]. Another approach inspired by the evolving vector quantization (eVQ) for evolution and adaptation of clusters [32] is the so-called *FLEXFIS* family [34], in particular *FLEXFIS* for regression [33] and for classification [35] (referred to *FLEXFIS-Class*). The range of other alternative approaches includes ePL [28] (evolving participatory learning, deduced from Yager's participatory learning paradigm [49]), SAFIS [22] (a sequential learning algorithm), and evolving fuzzy neural networks such as SOFNN [27] or GDFNN [48].

All these methods have in common that they are life-long learning approaches, which means that they incorporate all data samples into the fuzzy models with equal weights and in the same order as they are coming in during the on-line process. Hence, older and newer information are treated equally, and fuzzy

[1] Regular International Conferences take place, e.g. IEEE Symposia such as EFS'06, Ambleside, UK; GEFS'08, Witten-Bommerholz, Germany; ESDIS-2009, Nashville, TN, USA, and numerous special sessions, tutorials and other activities that are dedicated to this emerging topic

models assign equal importance to all the samples seen so far. On-line model identification is advantageous, especially when convergence to an optimality criterion or stable state of the model structure can be achieved [34]. However, this only holds for data streams which are generated from the same underlying data distribution and do not show any *drift* or *shift* behavior to other parts of the input/output space [44]. *Drift* (respectively *shift*) indicates the necessity of (gradual) out-dating of previously learned relationships (in terms of structure and parameters) during the incremental learning process as they are not valid any longer and should hence be eliminated from the model (for instance, consider completely new types of images in a surface inspection system). An alternative to gradual out-dating is the concept of re-learning, which can be either done based on all samples seen so far, providing lower weights for older samples in the learning process, or based on the latest data blocks only. The first variant slows down the learning process significantly over time, such that on-line real-time demands are hardly met. The second variant has the problem that older data is usually completely forgotten when extracting the models based on the new data blocks from scratch, causing a crisp switch between two models (from the old to the new). With gradual forgetting, a smooth transition from an old model to a new one can be achieved instead of an abrupt switch. *Drift* handling (in connection with gradual forgetting) was already applied in other machine learning techniques, e.g. in connection with Support Vector Machines (SVMs) [25] [26], ensemble classifiers [39], and instance-based (lazy) learning approaches [14] [18]. However, to the best of our knowledge, this concept has not yet been applied to fuzzy systems (neither the concept of re-learning).

1.2 Our Approach

In this paper, we propose approaches to handling *drifts* and *shifts* using the example of EFS, incorporating *eTS* and *FLEXFIS*. However, parts of the concept, especially the detection of and reaction to *drifts* in the consequent parts of the rules, can be applied to a wider range of approaches. A more detailed description of *drifts*/*shifts* and specific data examples is given in Section 2. To develop an automatic approach, we investigated the following two issues:

(1) Detecting a *drift*/*shift* (Section 3): This includes the tracking of changes in the *age* of fuzzy rules during the on-line learning process. *utility* function [10] of fuzzy rules.
(2) Reacting to a *drift*/*shift* once it is detected (Section 4): 1.) in the antecedent parts of the rules, this is accomplished in *eClustering* (as applied in *eTS*) and in *eVQ* (as applied in *FLEXFIS*); and 2.) in the consequent parts of Takagi-Sugeno fuzzy systems (Section 4.2); this method is applicable to any approach exploiting TS fuzzy model architecture.

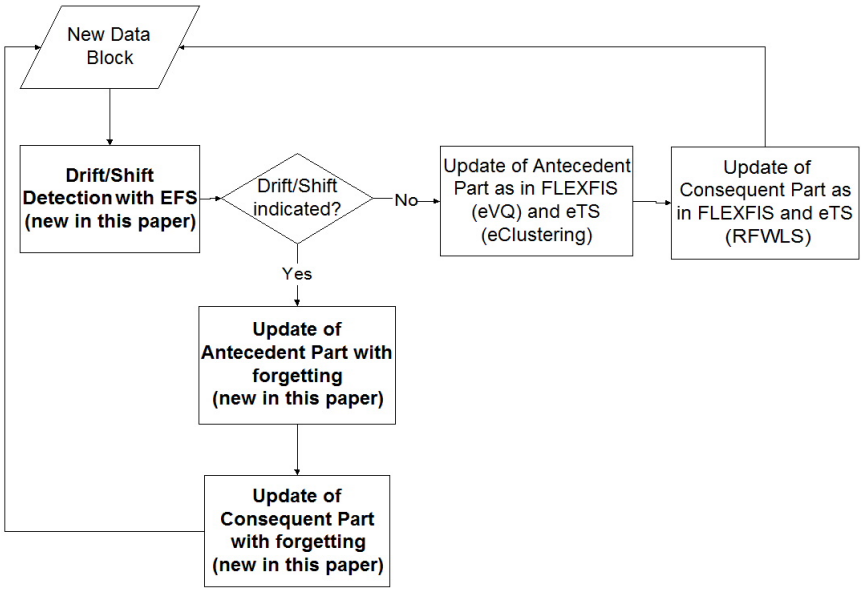

Fig. 1. Work-flow of using evolving fuzzy systems in connection with drift/shift handling techniques as proposed in this paper and marked in bold font

Figure 1 demonstrated an overview of the work-flow in an on-line modelling scenario, using evolving fuzzy systems connected with drift detection and reaction techniques as proposed in the subsequent section. The concepts marked with text in bold font are novel ones treated in this paper.

The paper concludes with an empirical evaluation of the impact of the proposed approaches on the predictive accuracy of the evolving fuzzy models when used with (on-line) real-world data sets in which *drifts* and *shifts* occur (Section 5). This includes 1.) on-line data from rolling mills, where the resistance value was to be predicted with high accuracy for each measurement and where different stitches may cause (slightly) different data distributions, indicating *drifts*; 2.) on-line data from surface inspection systems for CD imprints (various CD orders may indicate *drift*/*shift* in the data) and egg production, and 3.) and on-line data for modelling of the product composition in a distillation tower. Note that no benchmark data from the Internet or well-known data bases could be used for empirical evaluation, as these are usually all smooth, i.e. containing no *drifts*.

2 Problem Statement

In machine learning literature, they distinguish between different types of 'concept change' of the underlying distribution of (on-line) data streams: a) *drifts*, and b) *shifts*, see [44]. *Drift* refers to a *gradual* evolution of the concept over time. The concept *drift* concerns the way the data distribution slides

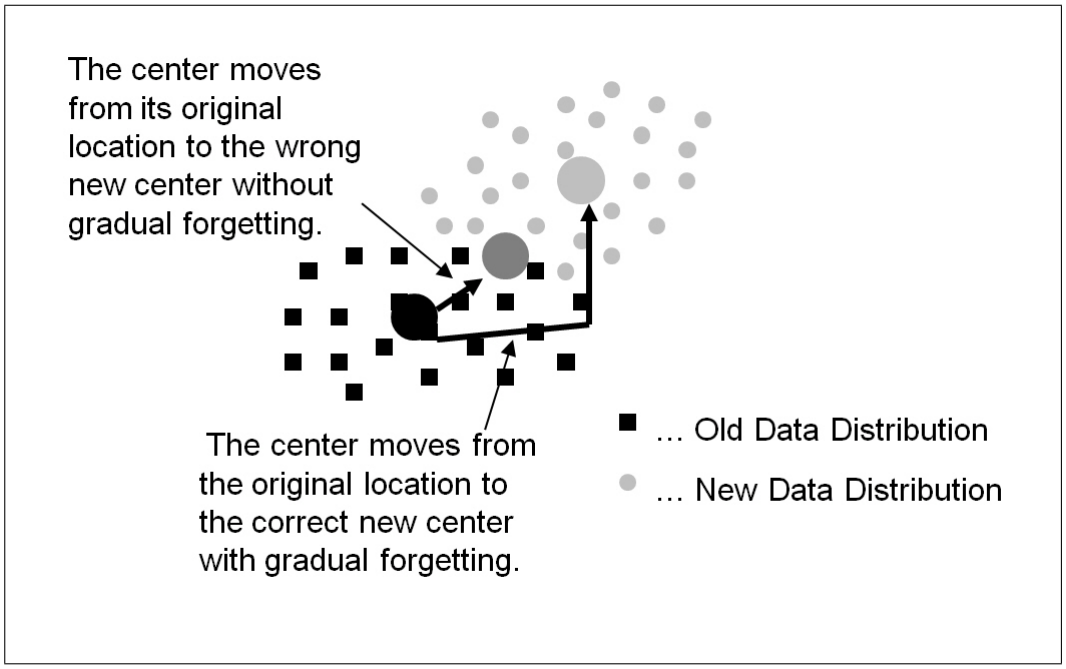

Fig. 2. A *drift* in an evolving cluster (used for learning the antecedent parts of EFS): Circles represent the underlying data distribution before the *drift*, squares represent the underlying data distribution after the *drift*; the big circles represent the cluster center of the original data (black), the center wrongly updated by a conventional learning algorithm (middle-grey) and the correct center of the new data distribution (light-grey)

smoothly through the data/feature space from one region to another. For instance, one may consider a data cluster moving from one position to another. This concept is closely related to the time-space representation of the data streams. While the concept of (data) density is represented in the data space domain, *drift* and *shift* are concepts in the joint data-time space domain. In machine learning literature they also recognize so called 'virtual concept drift' which represents a change in the distribution of the underlying data generating process instead of the change of the data concept itself [47]. Trends in time series data denote a specific form of drifts, where for instance periodic patterns (such as sinusoidal-type wave-forms) are drifting (slightly) away from the x-axis. In the joint feature space, this may cause drifting regions or clusters as visualized in Figure 2 (an artificial example, in fact). There, the original data distribution (in a 2-D data space) is marked by circular samples, which changes over time into a data distribution marked by rectangular samples. If a conventional clustering process would be applied that weights all new incoming samples equally, the cluster center would end up exactly in the middle of the combined data cloud, averaging old and new data.

To address the *drift* problem (or trends) in data streams, a mechanism is to be triggered once such a *drift* is detected. This may involve an incremental learning mechanism that assigns higher weights to new than to old data. This is

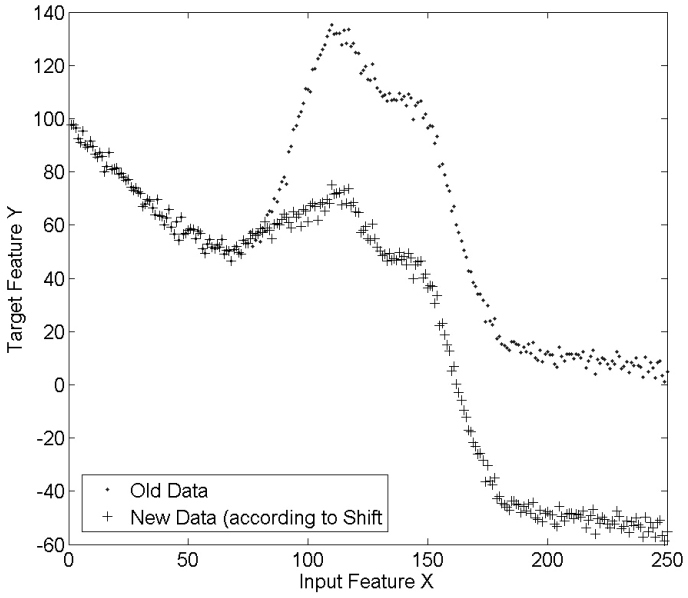

Fig. 3. Example of a *shift* in the output variable; compare the dots (original data distribution) with the crosses (data distribution after the *shift*) on the right-hand side of the graph

accomplished by applying a gradual forgetting process in order to guarantee a smooth transition from the old state (before the *drift*) to the new one (after the *drift*) and is described in Sections 4.1.1 and 4.1.2. The *shift* concept refers to an abrupt change in the underlying concept to be learned. A *shift* in the input space opens up a data cloud in an unexplored region and usually causes a new rule to be evolved by the evolving clustering algorithm. Such an occurrence can be caused by seasonality in data-streams, i.e. changing patterns occurring on different levels of one or more input features/variables (for instance, CO2 concentrations are tendentially higher in spring than in autumn), usually extending the input feature space (when seen as a joint high-dimensional space). In our evolving fuzzy systems approaches, such occurrences are handled automatically by extending the model to these new regions in the feature space through the evolution of a new (local) rule.

In Figure 3 a case of a specific *shift*, namely a shift in the output/target variable is shown, where an automatic evolution of new rules is not favorable: the original trajectory of the 2-dimensional non-linear relationship (consisting of noisy samples) is indicated with dots, whereas the *shift* is represented by the data samples marked as pluses forming a trajectory below the other one in the middle and right part of the image. In case the consequent parameters are adapted in the usual way using weighted recursive least squares (as in [13]) and old data is not forgotten, then the approximation curve of the fuzzy model lies exactly between these two trajectories. The same is the case when evolving separate rules for the shifted trajectory (causing two consequent functions,

one lying over the other). Also, gradual forgetting is recommended here as a smooth *drift* (where the surface slides softly from one position to another in vertical direction) may also occur instead of an abrupt *shift*. This implies the use of an adaptive forgetting factor dependent on the intensity of the *drift* (see Section 4.2).

In this paper, we demonstrate novel approaches for autonomous *drift* and *shift* detection and handling when using fuzzy rule-based systems of the Takagi-Sugeno type to learn from on-line data streams in an evolving manner [5]. We focus on EFS approaches exploiting the Takagi-Sugeno model architecture [43] but the *drift* detection and our technique for evolving the antecedent structure of the fuzzy rule-based methods are applicable more generally to any type of fuzzy model framework. Therefore, without limiting the scope of the validity of the conclusions, we consider the following fuzzy rule-based model assuming that p input features, C fuzzy rules and Gaussian membership functions describe the fuzzy sets, which are connected by the product t-norm in the antecedent parts defined as:

$$\hat{f}_{fuz}(\vec{x}) = \hat{y}_{fuz} = \sum_{i=1}^{C} l_i(\vec{x})\Psi_i(\vec{x}) \tag{1}$$

with the normalized membership functions

$$\Psi_i(\vec{x}) = \frac{\mu_i(\vec{x})}{\sum_{j=1}^{C} \mu_j(\vec{x})} \tag{2}$$

and μ_i as rule fulfillment of the ith rule

$$\mu_i(\vec{x}) = exp\left(-\frac{1}{2}\sum_{j=1}^{p}\frac{(x_j - c_{ij})^2}{\sigma_{ij}^2}\right) \tag{3}$$

and consequent functions

$$l_i(\vec{x}) = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + ... + w_{ip}x_p \tag{4}$$

where c_{ij} is the center and σ_{ij} the width of the Gaussian function appearing as a fuzzy set in the j-th antecedent part of the i-th rule.

3 Autonomous Detection of Drifts and Shifts in Data Streams Using EFS

This section describes the method for autonomous detection of *shifts* and *drifts* in data streams based on the *age* [1] of the cluster/fuzzy rule. This is an important step in the process of handling non-stationarity in data streams and

in building autonomous self-developing, self-learning, and evolving models and systems. Such a technique has already been applied to evolving self-learning and self-developing classifiers [8]. Here, we present a more detailed analysis.

Drifts occur when there is a significant difference between the distributions of the old and new data. In [26], *drift* is detected off-line using an SVM method [45]. Since we use fuzzy rule-based systems and a neuro-fuzzy model framework we need to develop an approach to 1.) autonomously detect *drifts/shifts* in the data concept on-line and 2.) to react on them in the system/model structure evolution in a recursive manner.

3.1 Detecting Drifts by the Age of Clusters (Rules)

In [1], the concept of cluster (respectively fuzzy rule) *age* was introduced, which is extended in this paper by including the $\Psi_{i,l}$ as the membership degree of the lth sample in the ith rule:

$$age^i = k - \frac{\sum_{l=1}^{n_i} I_l \Psi_{i,l}}{n_i} \tag{5}$$

where i is the rule index; n_i denotes the support of rule i; I_l denotes the time instance at which the data sample was read; k is the current time instance. Since $\sum_{l=1}^{n_i} I_l \Psi_{i,l}$ can also be accumulated recursively, the *age* can be calculated easily when necessary. The *age* of the cluster/rule changes with each new data sample being read. If the sample does not fall into that cluster (does not support that fuzzy rule), resp. supports the ith fuzzy rule with a low or even 0 value of $\Psi_{i,l}$, the *age* increases by the rate of one sample at a time. That means, if a cluster (fuzzy rule) is supported only slightly by any future samples after being initiated (say at time instance, t_i), then its *age* at any time instance, k will be approximately $k - t_i$. However, if a new data sample (between t_i and k) supports that fuzzy rule with a high value of $\Psi_{i,l}$, the *age* does not increase at the same rate, but at a smaller one; in other words, the cluster (fuzzy rule) is being *refreshed*.

Generally, the *age* of a cluster (fuzzy rule) has a range $[0, k]$. Figure 4 shows a curve representing age evolution - a process called *ageing*. Rules that are *older* are used and supported less by future data samples and become less important. The example depicted is based on a real application (propylene production data) and contains two *drifts*.

The cluster (fuzzy rule) *age* is important and closely linked to the data streams (which are sequences of data in time) and to the concept *drift*. We propose analyzing the *age* of clusters (respectively, fuzzy rules) on-line by using both, the gradient of the *ageing* curve and its second derivative, which indicate a

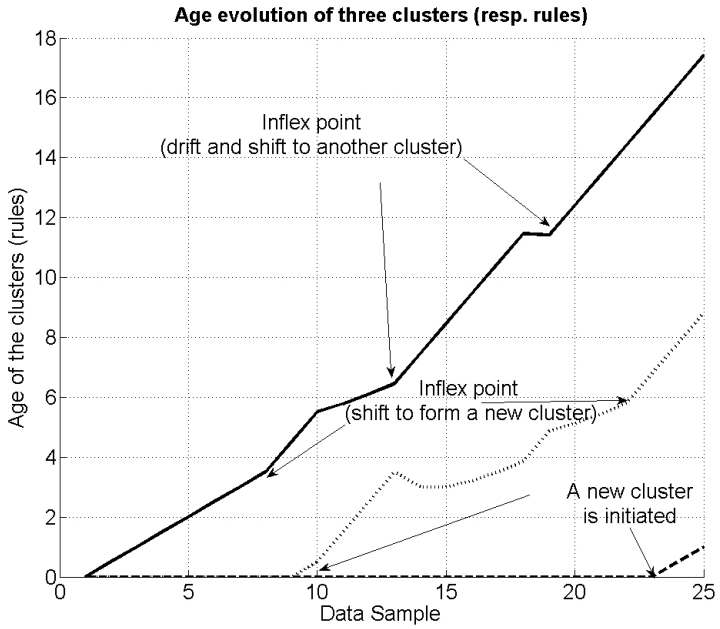

Fig. 4. A typical example of the development of rule *ages* from a real production system in which *shifts* and *drifts* occur (as outlined by the markers in the plots)

change in the slope of the *ageing* curve. Indeed, the rate of *ageing* is equivalent to the gradient of the *age* variable. We can evaluate the average *ageing* rate by calculating the mean value of the *age* variables and can then compare the current *ageing* rate to the mean *ageing* [10] to detect *drift/shift* in the data stream autonomously and on-line. When the change in *ageing*, and hence also in the slope, is significant, then obviously the second derivative of the *age* curve will be indicative of these inflection points. The right-hand side plot in Figure 5 shows an example in which the *drift* and activation phases are indicated in the right plot for a specific rule. In general, different rules may, of course, cause a different appearance of the corresponding *age* curves.

Please note that originally the concept of rule age was defined without including the membership degree Ψ, as used for deleting old rules from the rule base [1]. In case of smaller drifts as shown in Figure 2, the inclusion is necessary as otherwise the rule (cluster) is still always attached with data samples from the new distribution as still being the nearest cluster. Hence, no increase in the rule age curves would be visible. Instead, when using the membership degree, the rule age (of the ith rule) will increase as the numerator in (5) decreases due to the lower weights $\Psi_{i,l}$. In this sense, it is also possible to distinguish between *drift* and *shift* cases, as *drifts* will cause slight increases of the gradient in the rule *age* curves, and *shifts* more intense ones (as causing nearly 0 values of $\Psi_{i,l}$). In case of *shifts*, usually new rule/cluster centers are evolved not disturbing already generated clusters and occupying the new situation or other clusters are attached (shift away from one cluster often triggers as

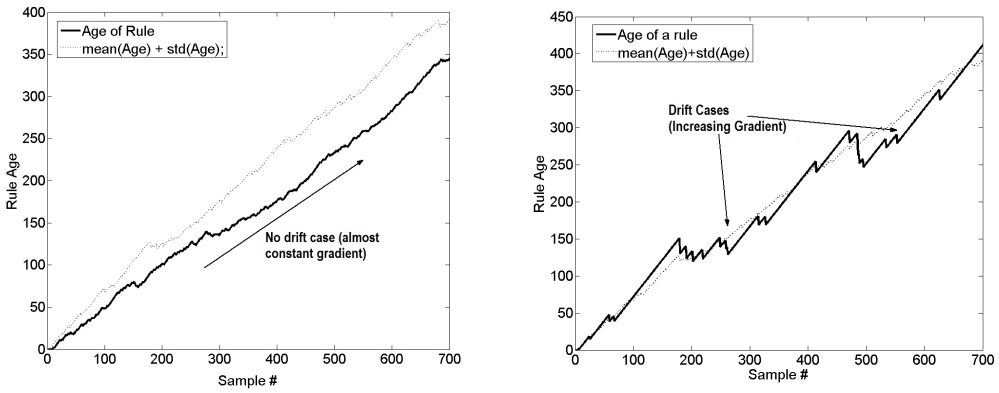

Fig. 5. Left: Evolving process where no *drift* for a specific cluster/rule occurs over time (x-axis); right: evolving process where two *drift* phases and two confirmation phases occur for a specific rule.

shift towards another cluster - as shown in Figure 4). In case of *drifts* the rule/cluster centers are forced to stronger movements in order to help them out from the converged situation within the old data distribution and being able to follow the new one (recall Figure 2). In the case when there is a shift only in the output variable, no new rule should be evolved, but the output trajectory of the function shifted (recall Figure 3), which can be forced by a forgetting approach in the rule consequent parameters (see Section 4.2). We can detect shifts of the output variable by virtually excluding it in the clustering process and see whether the input features alone are triggering a shift: if this is not the case, but it happens when the output is included (as usually done in the incremental product space clustering methods in our EFS approaches, *eClustering* in *eTS* and *eVQ* in *FLEXFIS*), then obviously a shift in the output variable takes place.

An alternative approach for discriminating between *drifts* and *shifts* is simply by calculating the rule *ages* just over the number of samples belonging to a certain rule (i.e. for which the rule was the nearest one) and not over all samples seen so far. Then, an increase in the gradient of the *age* curves always indicates a *drift*; *shift* cases (in the output) have to be treated separately as mentioned above.

4 Reactions by EFS to Drifts and Shifts in Data Streams

4.1 Reaction to Drifts and Shifts in the Antecedents

In this section, we introduce methods for addressing both *drifts* and *shifts* in data streams by adapting learning mechanisms for consequent parts and by

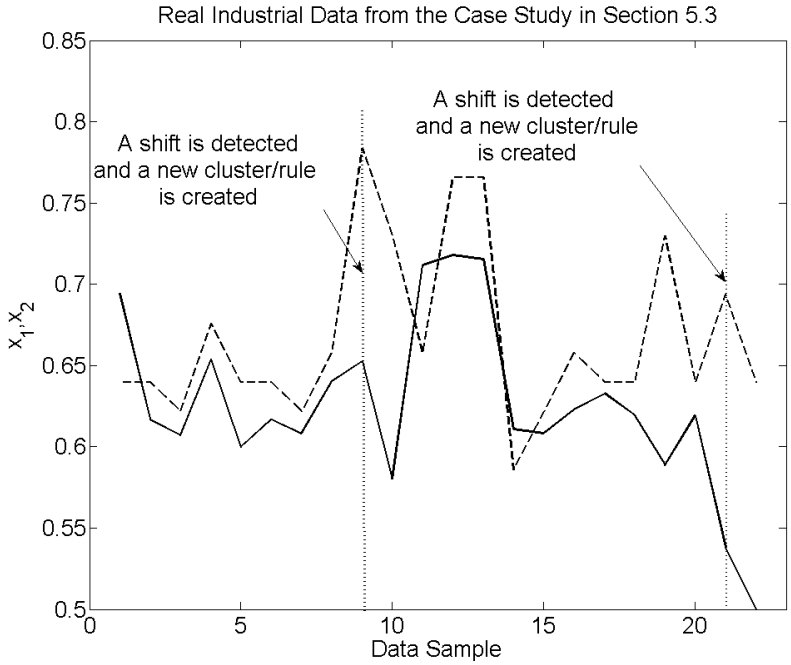

Fig. 6. *Shift* in the time domain representation and two new rules evolved as a reaction to it.

evolving antecedent parts. We demonstrate this using the example of the popular *eTS* method [5] and on *FLEXFIS* [33]. Whereas our method of reacting to *drifts* and *shifts* in the consequents can be applied to a broader range of EFS approaches (note that many of these approaches apply (weighted) recursive least squares (wRLS) learning methods), our approach to reacting to *shift* and *drift* in the antecedent parts is specifically tailored to the clustering approaches used - in our case these are *eClustering* [3] for *eTS* and *eVQ* [32] for *FLEXFIS*.

4.1.1 Reacting to Drifts and Shifts in Data Streams in eClustering

In *eTS*, *shifts* in data streams are detected quite naturally by recursive online calculation of the global data density/potential/mountain function [5] at the current data sample, and comparing this to the data density at all focal points (cluster centers) existing so far in the fuzzy rule base. Reaction to a detected *shift* is either by a) forming a new rule around a new data sample, which becomes an attraction point for the global data distribution (also see Figure 6), or b) replacing a fuzzy rule which in itself consists of; i) forming a new rule around the new point — as in the previous step, and ii) removing the rule which has lower density and is close to this newly added one [2]. When a *shift* in the data concept is detected, the respective fuzzy rules are being removed/deleted or new ones are added. If locally optimal learning is being applied, then removing a cluster, and hence a locally valid Kalman

filter/RLS, does not affect the overall learning significantly (only through the fuzzy weight Ψ [5]). If globally optimal learning is being applied, removing a cluster (respectively fuzzy rules) affects n columns and n rows of the covariance matrix directly and the remaining values of the covariance matrix indirectly [5]. A significant delay in reacting on *drifts* may arise as an evolution/re-setting of an old cluster center requires a higher density (number) of samples appearing in a new data cloud than in an already existing one. This means, that in the case when the data distribution slides from an old (heavy) local region to another, it may take quite a long time that a sample in the new data distribution has a higher potential than the older cluster center. Hence, in the case of a detected *drift*, the potential of this cluster center where the *drift* occurs is re-set to a small value, forcing an earlier evolution/re-setting of centers after the detection of the *drift*.

4.1.2 Reacting to Drifts and Shifts in eVQ

When *shifts* occur in the data stream, new clusters are usually evolved automatically in *eVQ*, because *shifts* cause new data clouds in explored regions, at least in regions further away than a fraction of the space diagonal in the input space (to which the vigilance parameter ρ is usually set, i.e. 0.1 to 0.3 of the space diagonal, see [32]).

We define the tracking concepts for the ith rule throughout this section, which can be generalized to any rule in a straightforward manner. To react to *drift* in the antecedent parts of the rules for *FLEXFIS*, we propose re-adjusting the parameter in the *eVQ* clustering algorithm [32] that steers the learning gain and hence the convergence of the centers and surfaces of the cluster over time (with more samples loaded), namely η. This convergence is according to a specific adjustment of the learning gain in the same way as also exploited in conventional VQ, which was proven to converge to the k-means solution [21] obtained by solving a least squares optimization problem between data samples and nearest centers [20]; following this approach, we have for the ith cluster:

$$\eta_i = \frac{0.5}{n_i} \tag{6}$$

where n_i the number of samples forming the cluster (i.e. for which the cluster was the nearest one) in the past; therefore, for different clusters we obtain different (decreasing) learning gains. The is applied in the update of the prototype c_i of the ith cluster in the following way:

$$\vec{c}_i^{(new)} = \vec{c}_i^{(old)} + \eta_i(\vec{x} - \vec{c}_i^{(old)}) \tag{7}$$

Thus, the old center $\vec{c}_i^{(old)}$ is moved towards the current sample ($\vec{x}$) by a fraction of the distance between the current sample and its center coordinates, achieving a new center $\vec{c}_i^{(new)}$. Considering equation (6), it becomes clear that

the learning gain decreases with the number of samples forming the ith rule (n_i), which is a favorable characteristic in order to converge to 1.) the center of a local region (data cloud) [32] and 2.) to optimality in the least squares sense when updating consequent parameters in the Takagi-Sugeno fuzzy systems [31]. If the learning gain would not be decreased, fluctuation of the cluster centers and therefore of the antecedent parts of the rules would ensue. Hence, the whole EFS approach would become unstable/lose stability. However, if a *drift* occurs in a data stream, the effect described above is no longer desirable, because center(s) and width(s) of the cluster(s) should adapt to the new data distribution (as shown in Figure 2). Unlike a *shift*, a *drift* usually does not trigger a new cluster, just indicates a stronger movement of already existing clusters as expected. To re-activate the converged clusters, i.e. re-animate them for stronger movements in a drifting case, we force a sudden increase of the learning gain for the first sample in the *drift* phase, followed by a gradual decrease for the next samples in order to balance in the new sample distribution in the same manner as is done for original ones.

4.1.2.1 *Drift* Tracking with Abrupt Increase of the Learning Gain Followed by Gradual Decrease Here, we propose the following mechanisms for the learning gain η: first, we transform the forgetting factor λ, which is used for gradual out-weighting in consequent learning (see (25)), and assign a value in $[0, 1]$ to the intensity of a *drift*. We map 0.9 (the minimal value for λ) to 0.99, while 1 is mapped to 0. Hence, we obtain:

$$\lambda\_trans = -9.9\lambda + 9.9 \tag{8}$$

When a *drift* occurs, we then re-set the number of samples forming the ith cluster (n_i) (used in the denominator of the calculation of η_i) according to

$$n_i = n_i - n_i * \lambda\_trans \tag{9}$$

This means that the stronger the *drift* is, the more n_i is decreased and hence the stronger the forgetting effect will be. Figure 7 shows how η_i develops (lines) in usual (i.e., non-*drift*) case (for the first 100 samples). Then, a drifting scenario was triggered at three intensity levels, leading to three λ values. The development curves of η_i in these three cases are shown as a solid line, dashed line and dashed dotted line. After the *drift* indicator (at sample 100), the learning gain is decreased in usual way allowing the center to converge to the new data distribution (in the same manner as is the case in eVQ using (6) for convergence reasons, see explanation above).

This resetting of n_i to a significantly lower value also affects the update of the ranges of influence of the clusters, as these are estimated using the recursive

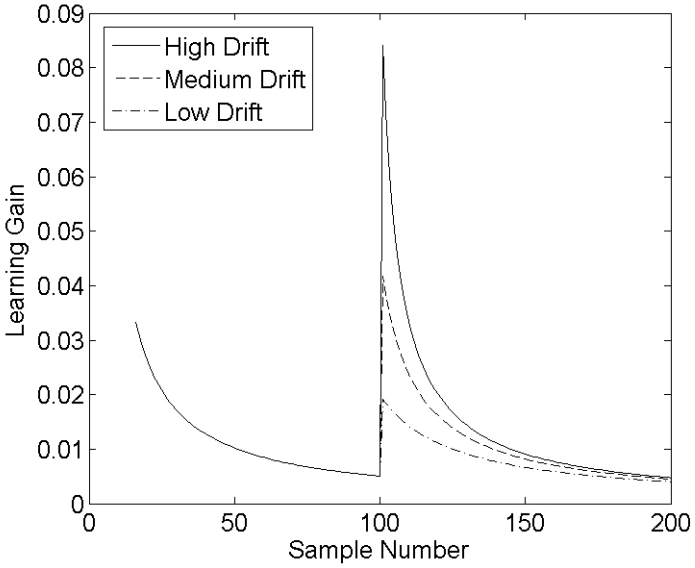

Fig. 7. Abrupt increase in the learning gain η in case of a *drift* (after 100 samples) when using different values for the forgetting factor λ (indicated by different line styles).

variance formula dimension-wise (the variance of the data samples forming a cluster is calculated separately for each dimension):

$$\sigma_{i,j}^2(new) = \\ \frac{n_i - 1}{n_i}\sigma_{i,j}^2(old) + \Delta c_{i,j}^2 + \frac{1}{n_i}(c_{i,j} - x_j)^2 \quad \forall j = 1, ..., p+1 \tag{10}$$

with p as the number of inputs (note that we perform a product-space clustering, so inputs and output feature in a joint space), $\sigma_{i,j}^2(old)$ the old variance of the ith cluster with respect to the jth dimension, $c_{i,j}$ the center of the ith cluster in the jth dimension, x_j the jth dimension in the new data sample and $\Delta c_{i,j}$ the difference of the old cluster center to the new one (after updating with (7)) with respect to dimension j. The ranges of influence are needed to obtain the widths $\sigma_{i,j}$ of the Gaussian fuzzy sets as used in the Takagi-Sugeno fuzzy systems, see (3). Obviously, when n_i is reset to a low value according to (9), the update of $\sigma_{i,j}$ is revived because 1.) the influence of $\sigma_{i,j}^2(old)$ decreases significantly (the smaller n_i is, the lower $\frac{n_i-1}{n_i}$ gets) and 2.) the deviation of the new sample to the center (last term in formula) gets significantly higher according to (7). This means that new samples in the new data distribution are represented in $\sigma_{i,j}$ with a much stronger influence than the older ones in the old data distribution, thus achieving a forgetting effect.

Figure 8 depicts a two-dimensional clustering example with a data drift from one local region (data cloud in the left image, samples indicated as dots) to a neighboring one (additional cloud in the right image, samples indicated as pluses). In the upper row the trajectories of cluster centers (big dark dots) and

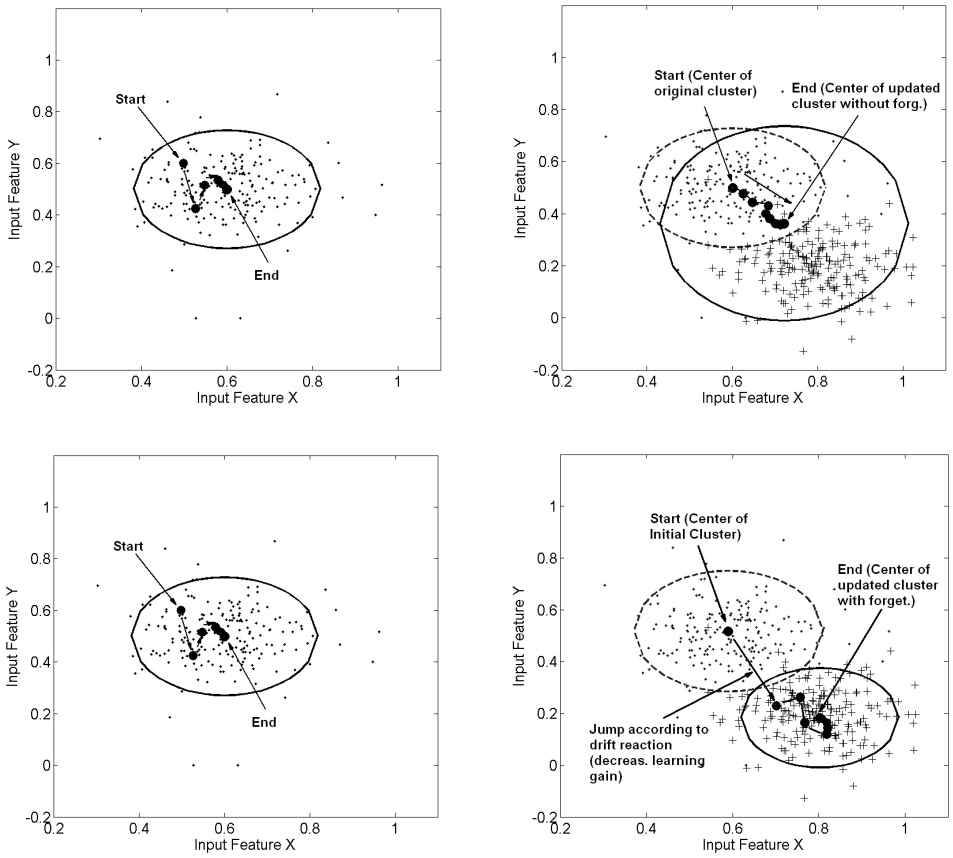

Fig. 8. Left: original data cloud; right: drifted data cloud indicated by crosses; the upper row shows the trajectory of cluster movements without abrupt decrease in the learning gain (no forgetting), the lower row with abrupt decrease (forgetting)

the range of influence (ellipsoid in solid line) of the final cluster is shown for conventional adaptation of cluster centers without forgetting. The bottom two images in Figure 8 illustrate the analogous case using forgetting as described above. Note, in this case, the bigger jump from the converged cluster center (as shown left image) to the new data distribution is bigger, forcing a new compact cluster for the new data distribution to emerge (marked with crosses). Without forgetting, an incorrect big joint cluster is created, because all samples are weighted equally (see right image, upper row).

4.2 Reaction to Drifts and Shifts in the Consequents

For the rule consequents, *drifts* and *shifts* can be handled in one sweep, since only an approximate setting of the forgetting factor value is required as described below. Whenever a *drift* in the output variable occurs (see Figure 3) and is detected by the approach described in Section 3, it is necessary to apply a specific mechanism to the sample-wise incremental learning steps of the

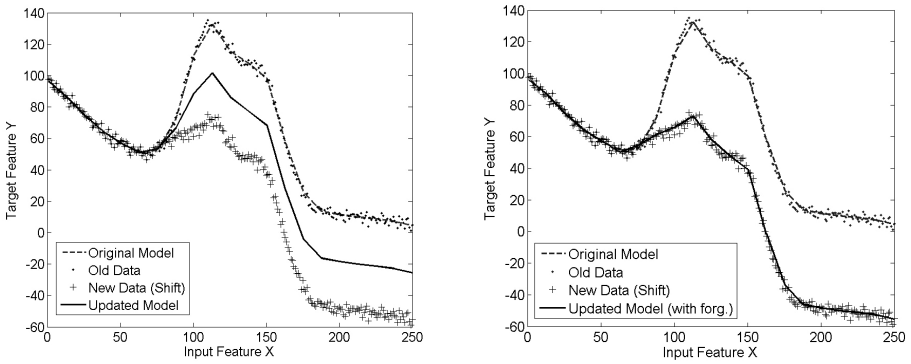

Fig. 9. Left: The adapted model (dotted line) and the new incoming samples (dark dots) when applying a conventional recursive weighted least squares approach as defined in (11)-(13); right: the adapted model (dotted line) when including a forgetting factor of 0.95 and using (21)-(23); the approximation surface coincides exactly with the trajectory of the data samples representing the new data distribution

consequent parameters in Takagi-Sugeno fuzzy systems as defined in (1).

The usual incremental learning approach exploits local learning of the consequent functions (which has some advantages over global learning, particularly providing much more flexibility for adjoining and deleting rules on demand, see [5] [10] [31] [8]) and defines a recursive fuzzily weighted update scheme, in which the membership degrees of the fuzzy rules serve as weights. For the ith rule, the update scheme is defined in the following way:

$$\hat{\vec{w}}_i(k+1) = \hat{\vec{w}}_i(k) + \gamma(k)(y(k+1) - \vec{r}^T(k+1)\hat{\vec{w}}_i(k)) \tag{11}$$

$$\gamma(k) = \frac{P_i(k)\vec{r}(k+1)}{\frac{1}{\Psi_i(\vec{x}(k+1))} + \vec{r}^T(k+1)P_i(k)\vec{r}(k+1)} \tag{12}$$

$$P_i(k+1) = (I - \gamma(k)\vec{r}^T(k+1))P_i(k) \tag{13}$$

with $P_i(k) = (R_i(k)^T Q_i(k) R_i(k))^{-1}$ the inverse Hessian matrix at the previous time instance k, $\vec{r}(k+1) = [1 \;\; x_1(k+1) \;\; x_2(k+1) \;\; \ldots \;\; x_p(k+1)]^T$ the regressor values of the $k+1$th data sample, and Ψ_i as the fulfillment degree of the ith rule, serving as weight in the recursive least squares algorithm, $y(k+1)$ the measured value of the output (target) variable in the new time instance $k+1$ and $\hat{\vec{w}}_i(k)$ the consequent parameter vector of the ith rule as defined in (4) estimated by the first k samples.

This approach indeed includes different weights for different samples (see the Ψ_i in the denominator of (12)), but rather than being motivated by a changing characteristics of the sample distribution over time, this relates to the local position of the samples with respect to the rules. This means that all newer samples at the same local positions relative to the rules as the older samples are included with the same rule weights in the update process. For the drift problem as shown in Figure 3, the wRLS estimation hence results in a fuzzy

model approximation curve as shown in the left plot of Figure 9 with dotted lines. Of course, the approximation lies precisely between the two trajectories, since it seeks to minimize the quadratic errors (least squares) of all samples to the curve. Hence, it is necessary to include a parameter in the update process, which forces older samples to be out-dated over time. Gradualism is important here in order to guarantee a smooth forgetting and to prevent abrupt changes in the approximation surface. For this purpose, we re-define the least squares optimization function for the ith rule by

$$J_i = \sum_{k=1}^{N} \lambda^{N-k} \Psi_i(\vec{x}(k)) e_i^2(k) \longrightarrow \min_{\vec{w}} \tag{14}$$

with $e_i(k) = y(k) - \hat{y}(k)$ the error of the ith rule in sample k and λ a forgetting factor. Assuming that N is the number of samples loaded so far, this function outdates the sample processed i steps before by λ^{N-k}. Common values of λ lie between 0.9 and 1, where a value near 1 means slow forgetting and a value near 0.9 means fast forgetting of previously loaded data samples. The exact choice depends strongly on the behavior of the *drift* (see below). From (14), it is quite clear that the weighting matrix for rule i becomes

$$Q_i = \begin{bmatrix} \lambda^{N-1}\Psi_i(\vec{x}(1)) & 0 & ... & 0 \\ 0 & \lambda^{N-2}\Psi_i(\vec{x}(2)) & ... & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & ... & \lambda^0\Psi_i(\vec{x}(N)) \end{bmatrix}$$

with N the number of data samples and $\Psi_i(\vec{x}(j)$ the membership degree of sample j to the ith rule. Let the weighted error vector for the ith rule at time instance k be defined as

$$e_{k,i} = [e(1)\sqrt{\lambda}^{k-1}\sqrt{\Psi_{1,i}} \quad \dots e(k)\sqrt{\Psi_{k,i}}]^T \tag{15}$$

with $\Psi_{k,i} = \Psi_i(\vec{x}(k))$ and the objective function J_i for the ith rule at time instance k defined as

$$J_{k,i} = e_{k,i}^T e_{k,i} \tag{16}$$

For the $(k+1)$-th data set, we define:

$$J_{k+1,i} = \sum_{j=1}^{k+1} \lambda^{k+1-j} \Psi_{j,i} e_i^2(j) \tag{17}$$

respectively:

$$J_{k+1,i} = \lambda \sum_{j=1}^{k} \lambda^{k-j} \Psi_{j,i} e_i^2(j) + \Psi_{k+1,i} e_i^2(k+1) = \lambda e_{k,i}^T e_{k,i} + e_i^2(k+1) \tag{18}$$

or

$$J_{k+1,i} = \begin{pmatrix} \sqrt{\lambda}\Psi_{k,i} e_{k,i} \\ \Psi_{k+1,i} e_i(k+1) \end{pmatrix}^T \begin{pmatrix} \sqrt{\lambda}\Psi_{k,i} e_{k,i} \\ \Psi_{k+1,i} e_i(k+1) \end{pmatrix} \tag{19}$$

Recall the conventional LS estimator at time instance k, which minimizes $J_{k,i} = \sum_{j=1}^{k} \Psi_{j,i} e_i^2(j)$, i.e. $\hat{\vec{w}}_i(k) = (R_i(k)^T Q_i(k) R_i(k))^{-1} R_i(k)^T Q_i(k) y(k)$ [36]. The modified wLS solution with forgetting at step $k+1$ is:

$$\hat{\vec{w}}_i(k+1) = \left(\begin{pmatrix} \sqrt{\lambda}\sqrt{Q_i(k)} R_i(k) \\ \sqrt{Q_i(k+1)} \vec{r}_i^T(k+1) \end{pmatrix}^T \begin{pmatrix} \sqrt{\lambda}\sqrt{Q_i(k)} R_i(k) \\ \sqrt{\Psi_i(k+1)} \vec{r}_i^T(k+1) \end{pmatrix} \right)^{-1} \begin{pmatrix} \sqrt{\lambda} Q_i(k) R_i(k) \\ \Psi_i(k+1) \vec{r}_i^T(k+1) \end{pmatrix}^T \begin{pmatrix} \sqrt{\lambda}\vec{y}(k) \\ y(k+1) \end{pmatrix} \tag{20}$$

Following the same recursive deduction scheme as in conventional wLS [29] (defining $P(k+1)$ as the inverse matrix in (20), exploiting the fact that $P_i(k+1) = (P_i^{-1}(k) + \vec{r}_i(k+1)\Psi_i(k+1)\vec{r}_i^T(k+1))^{-1}$, and applying the matrix inversion theorem, also known as Sherman-Morrison formula [41]), we obtain the following incremental update formulas:

$$\hat{\vec{w}}_i(k+1) = \hat{\vec{w}}_i(k) + \gamma(k)(y(k+1) - \vec{r}^T(k+1)\hat{\vec{w}}_i(k)) \tag{21}$$

$$\gamma(k) = \frac{P_i(k)\vec{r}(k+1)}{\frac{\lambda}{\Psi_i(\vec{x}(k+1))} + \vec{r}^T(k+1) P_i(k) \vec{r}(k+1)} \tag{22}$$

$$P_i(k+1) = (I - \gamma(k)\vec{r}^T(k+1)) P_i(k) \frac{1}{\lambda} \tag{23}$$

Now, the final question is how to set the parameter λ in order to guarantee an appropriate *drift* tracking. Two possible solutions exist:

(1) Define a fixed parameter value of λ in [0,1] according to some *prior* knowledge about how strong the *drifts* might become.
(2) Introduce a variable forgetting factor, which depends on the intensity/speed of the *drift*.

For the latter, methods exist that include the gradient LS error with respect to λ to track the dynamic properties (see, e.g. [42]). In this paper, we propose a strategy to deduce it directly from the age curves analysis (Section 3.1), because it provides the degree of change in the rules *ages*, which indicates the speed of the *drift* (see Section 3). In Section 3, we mentioned that the *age* of a rule always lies in $[0, k]$. Hence, we normalize the *age* of the ith rule to $[0, 1]$ by

$$age_i\_norm = \frac{age_i}{k} \tag{24}$$

in order to achieve gradients of the normalized rule ages Δage_i_norm also lying in $[0, 1]$. Based on the figures in Section 3, we explained that a sudden increase in the gradient of the rule *age* curves indicates a *drift* in the data stream. This means, whenever the change of the gradient is significant, wRLS with forgetting should be triggered. We use the following estimation for λ:

$$\lambda = 1 - 0.1\Delta^2 age_i\_norm \tag{25}$$

with $\Delta^2 age_i\_norm$ the second deviation of the *age* (curve) of the ith rule.

This guarantees a value of λ between 0.9 (strong forgetting) and 1 (no forgetting), according to the degree of the gradient change (1 = maximal change, 0 = no change) (note: to the best of our knowledge, a value smaller than 0.9 produces instabilities in the models). The forgetting factor is then kept at this level for a while, as otherwise a single sample would cause a gradual forgetting. We reset λ to 1, when a stable gradient phase is achieved (usually after around 20 to 30 samples with moderate $\Delta^2 age_i\_norm$ values). Resetting λ to 1 is necessary to prevent forgetting from continuing in the new data distribution. This causes the *drift* phase in the antecedents to stop. In the case of a *shift*, we always set λ to the minimal value of 0.9. Another important issue is concerned with the consequences of over-detecting *drifts* or *shifts*. In this case, older samples are forgotten and the significance of the parameters and the whole model decreases. However, the surface of the model stays at the correct (old) position in the detected *drift/shift* region, since the samples do not move (significantly). In other regions the update will be quite small. However, it may become significant with more samples loaded, since all rules always fire to a small degree (as Gaussian fuzzy sets possess infinite support). This could cause an unlearning effect [30]. Hence, keeping the *drift* phase low at 20 to 30 samples is very important.

5 Evaluation

This section deals with the evaluation of the impact of reacting to *drifts* and *shifts* in the case of data streams in which *drifts* and *shifts* actually occur. This was done by implementing the approaches discussed throughout this paper in

the two EFS approaches (*eTS* and *FLEXFIS*). One application example was concerned with identifying a prediction model on-line for the resistance value of steel plate in a rolling mill. The second came from a surface inspection system supervising CD imprints and eggs with respect to faults. The third application example came from the chemical industry, where *eTS* was applied for modeling and predicting the chemical product composition by modelling the product composition in a distillation tower.

5.1 On-Line Prediction Models for Rolling Mills

The task was to identify a prediction model for the resistance value of a steel plate in a rolling mill. In a first step, we used some off-line (pre-collected) measurement data in order to obtain a rough idea about the quality the fuzzy model achieved. Then we refined the prediction model with newly recorded on-line data. The second step was possible, because first a prediction for the resistance was provided, influencing the whole process of the rolling mill, whereas a few seconds later (after passing the steel plate), the real value of the resistance was measured, which was then incorporated into the model adaptation process. Thus, not the predicted (and maybe false) value was used for further adaptation, but the correct, measured value; hence, the method can therefore be considered as a potential procedure for improving the models. The initial situation was as follows: an analytical model in which some parameters were estimated through linear regression and should be improved by the fuzzy model component *FLEXFIS*, which was applied purely on the basis of measurement data. The analytical model is physically motivated by material laws formulated by Spittel and Hensel [23]. Using a modified form these laws, the resistance value can be estimated by an exponential function of the temperature, speed and thickness of the steel plate. We achieved a significant improvement in the predictive power of the analytical models using fuzzy models trained with *FLEXFIS* in off-line batch mode. The data set for training and cross-validation comprised 6000 samples, and 6600 test samples were used to estimate the predictive power for new on-line samples. A further improvement could be achieved by updating the fuzzy models during on-line operation mode. For details of the experimental setup and results see [33] (and also Table 1, which summarizes all the results).

Subsequently, we wanted to examine whether reacting to *drifts* by gradually forgetting older samples during the on-line process further improves the quality of the models. The application of a drift reaction method was justified by the fact that the operation process of rolling mills is divided into different "stitches". One stitch represents one closed cycle in the rolling process. In on-line mode, the measurements are taken continuously from stitch to stitch. For the current stitch, the previous stitch should be of little or even no importance.

Table 1
Comparison of evolving fuzzy prediction models obtained by conventional *FLEXFIS* and when applying gradual forgetting as outlined in Sections 4.1 and 4.2

Method	**MAE**	**Max MAE Too High / Max MAE Too Low / # MAEs** > 20
Analytical	7.84	63.47 / 87.37 / 259
Static fuzzy models	6.76	45.05 / 81.65 / 176
FLEXFIS	5.41	38.05 / 78.88 / 159
FLEXFIS with grad. forgetting	4.65	31.99 / 74 / 68

However, the measurements from the previous stitch are already included in the fuzzy models as they are being updated by the current samples. This means that older samples from the previous stitch should be forgotten when including samples from the current stitch. Furthermore, no *drift/shift* detection is needed, as the *drift/shift* is indicated by the beginning of a new stitch. This start signal was transferred to the computer and was also included in the stored measurement data. As no *drift* detection was carried out, λ was set to 0.97, which is a good compromise between fast forgetting (=strong locality of models) and slow forgetting (=weak locality of models). The results are summarized in Table 1. It shows that both the static and the evolving fuzzy model outperformed the analytical model in predictive accuracy. Three different types of errors were reported: 1.) the mean absolute error over all won-line samples (note that first a prediction was made and then the model updated with the same samples and based on feedback), 2.) the number of mean absolute error (MAE) greater than 20, 3.) the mean MAE over all samples for which the prediction was too low, and 4.) the mean MAE over all samples where the prediction was too high. The last value is the most important one as it implies that the steel plate may get damaged, whereas predicting too low values has no such detrimental effects. The essential row in Table 1 is the last one, which shows the impact of gradual forgetting when triggered at each new stitch — compare to *FLEXFIS* without the forgetting procedure: the number of predictions for which the errors were too high could be reduced by more than 50% and the maximal error was lowered to almost 31.99, which was quite beneficial. Also, the advantage of evolving the fuzzy models during the on-line process (four more rules) is clearly evident in Table 1 (compare the second with the third and the fourth row).

Another interesting aspect is that the error of individual measurements starts to *drift* over time when gradual forgetting is not applied. This is visualized in the left image of Figure 10, which shows the individual errors over the 6600 on-line samples: note the drift of the main error area away from the zero line

at the end of the data stream. The right-hand side plot shows the individual errors when applying gradual forgetting: the *drift* occurrence at the end of the plot could be eliminated.

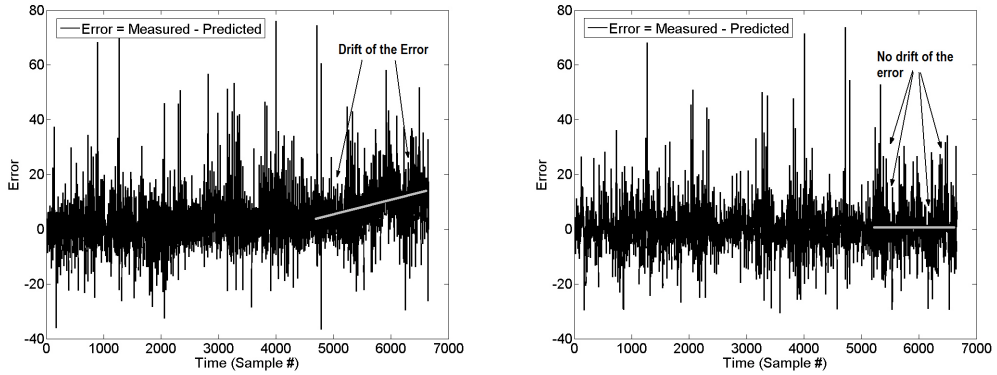

Fig. 10. Left: The error curve for the 6600 on-line samples when no forgetting is applied: at the end the error starts drifting as the body area around zero increases to a value around 10 to 15; right: no drift in case of gradual forgetting as applied at the beginning of each stitch.

5.2 Surface Inspection of CD Imprints and Eggs

The task was to automatically detect errors in CD imprints to sort out bad ones during the production process. Errors include color *drift* during offset print, a pinhole caused by a dirty sieve (color cannot pass through), occurrence of colors on dirt, and palettes running out of ink. For this purpose, a camera was installed directly above the conveyor belt over the trays which record the CDs. The images were then processed further using an on-line image classification framework [40] that was setup for an EU-Project (DynaVis — see www.dynavis.org), and classifies the samples into good and bad ones. A fault-free master was used to obtain a contrast image in which potential CD imprint faults could be identified. The most difficult challenge was to distinguish between real errors and pseudo-errors in these imprints (a pseudo error, for example, a small *shift* of the CD in the tray that causes an arc-type or circular deviation region in the contrast image). The contrast image was processed further by extracting a high-dimensional feature set characterizing the deviation pixels in these images. Based on the feature set extracted from a pre-labelled training data set, various machine learning methods (CART, C4.5, k-NN, SVMs, eVQ-Class, ...) were used to train classifiers, and also fuzzy classifiers with the help of *FLEXFIS-Class*, the classification variant of *FLEXFIS* (see [40] for a comparison of methods) were employed.

This had to be further refined with on-line data samples, as the initial setup

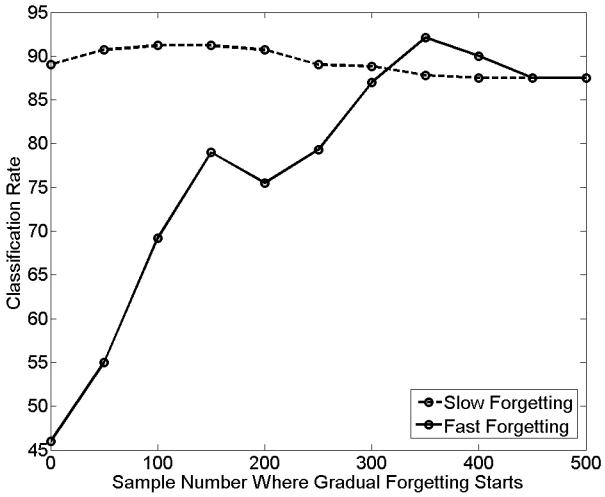

Fig. 11. *Drift* analysis of the CD Imprint data labeled by Operator 2, applying two different speeds in the gradual forgetting process and starting the '*drift* phase tracking' at different sample numbers (x-axis); the y-axis shows the accuracy for the separate test set.

was quite time-consuming for the experts. Especially, the labeling process involved great workload effort, as the contrast images had to be labeled, in some cases even single objects in the images instead of complete images in order to guarantee high-performance classifiers [37]. This means that only a few training samples were labeled, which may cause over-fitting of the initial (off-line trained) classifiers, especially in high-dimensional cases. Hence, refining with more data during on-line mode is mandatory. For a set of 512 on-line samples, the conventional update scheme in *FLEXFIS-Class* achieved classification rates of 89.92%, increasing the classification rate by around 4% compared to static (initially built and not further adapted) classifiers.

Subsequently, we studied the impact of gradual forgetting in the antecedent and consequent part of the fuzzy models. It is important to note that *FLEXFIS-Class* evolves K Takagi-Sugeno fuzzy regression (sub-)models as defined in (1) for K different classes. Hence, the approaches proposed in this paper can be directly applied to all sub-models. The the importance of examining gradual forgetting is underlined by the fact, that the CD imprint data was originally recorded for different orders, where different orders may contain *drifts* or *shifts* in the image data distribution. We performed a *drift* analysis with different starting points (beginning with gradual evolution at 0, 50, 100, 150, ..., 500 on-line samples), with drift phases encompassing 30 samples (as suggested in Section 4.2) and with two different speeds for gradual forgetting. The results are shown in Figure 11, where the slow gradual forgetting is denoted by the dotted-dashed line and the fast gradual forgetting by the solid line. The sample number in the second half of the training set at which the *drift* is initialized is represented by the x-axis, the achieved accuracies on separate test data when

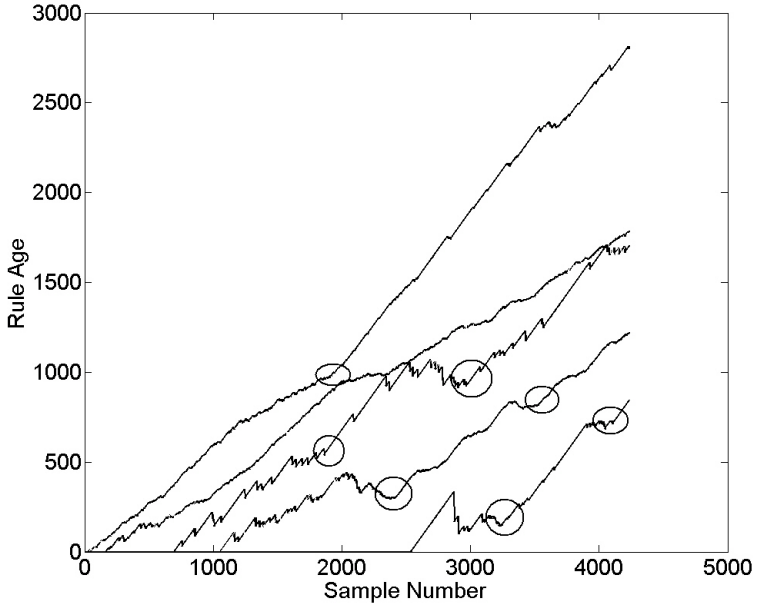

Fig. 12. Rule age curves for five rules evolved for the egg data set, containing 4238 samples and 17 input features, parts with significantly increasing gradients (drifts) marked by ellipses

including the gradual forgetting on the rest of the second half of the training data is represented by the y-axis. In fact, the accuracies were boosted up by about 3% (to 91.39% maximum) when initiating a *drift* phase at the correct sample number and choosing the appropriate speed in the gradual forgetting process. In our case, this means that a slow forgetting starting between samples 100 and 150 and fast forgetting starting at sample 350 is beneficial, as this tracks two different types of *drift*. In fact, combining these two results in a classification rate of over 92%, gaining another 1% accuracy.

Regarding the surface inspection scenario for eggs on the conveyor belt, the task was basically to distinguish between dirt and yolk occurrences, as both are causing significant entries in the deviation image (obtained by subtracting newly recorded image with a master). For our fuzzy classifiers evolved by *FLEXFIS-Class* (without drift detection and reaction), it was possible to achieve a classification rate of about 83%, which could be further increased to around 90%, when applying rule ages for drift detection and gradual forgetting of antecedents as explained in Section 4.1.2 for reacting onto the drift. The rule age curves of five rules evolved during the incremental learning phase are shown in Figure 12, the positions where drifts were detected indicated by ellipses.

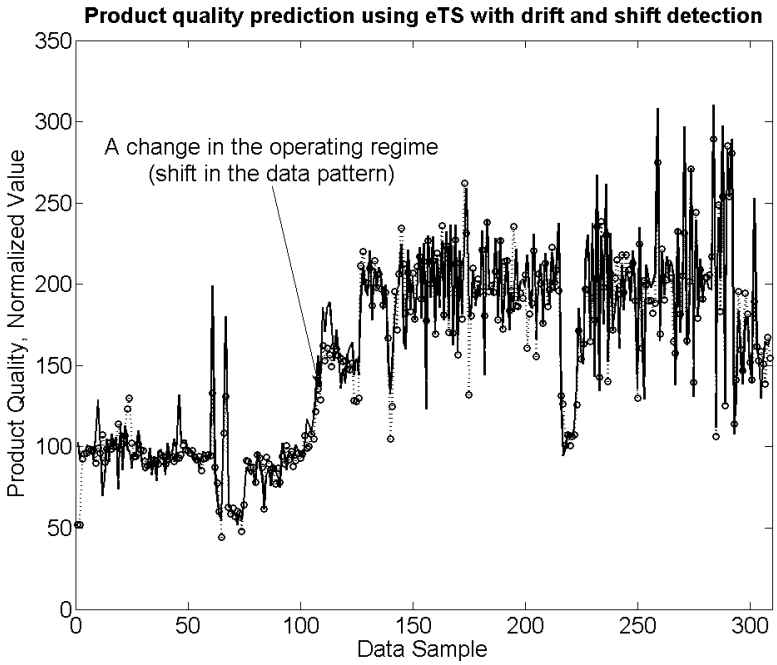

Fig. 13. Comparison of the predicted and real product quality data. Solid line - predictions by *eTS* with *shift* and *drift* detection; circles - real data.

5.3 Modeling the Product Composition in a Distillation Tower

A case study based on real data (courtesy of Dr. Arthur Kordon, The Dow Chemical Co.) from the chemical industry [7] was used to illustrate of the detection of and reaction to *drift* and *shift* in *eTS*. The *eTS* training method [9] was applied to model and predict the chemical product composition in a distillation tower. The traditional technique for estimating the various chemical properties of the compositions is based on off-line manual and cost-related laboratory analysis of the properties based on grab samples using gas chromatographs. The sampling period in the case of laboratory analysis is 8 hours and the accuracy is 2.2% measurement error ([7]). The data set included a change in the operating regime of the process, which brings a challenge imposed on the structure of the model (fuzzy rule based system). The data set also incorporated a number of other challenges, such as noise in the data, and a large number of initial variables. These problems cover a wide range of real issues in the industry. Process data retrieved from physical ('hard') sensors was used as inputs to the *eTS*, applying hourly averages for every eight hour period. The product composition (real output) was estimated by laboratory analysis for comparison (it see line with circles in Figure 13). Process data represent hourly averages around the time when the sample due to laboratory analysis has been taken. The total data samples includes 309 records.

The estimation of the product composition contained noise due to the nature of

Table 2
Accuracies obtained by *eTS* when applying drift detection and reaction (third column) and when performing conventional *eTS* (second column)

Error measure	**Without drift reaction**	**With drift reaction**	**Best (in theory) value**
NDEI	0.44278	0.3559	0
VAF, %	80.461	87.319	100
correlation	0.89971	0.9357	1

the analysis. A significant operating condition change took place after sample 127 (see Figure 13). If no detection of the shift in the data exists and no reaction to this shift in terms of changing structure of the underlying fuzzy rule-based model then the result is significantly poorer. Due to the detection and reaction to this shift in the data pattern (see Figures 4 and 6), the *eTS* rule-based system evolved its structure and achieved a better result (see Table 2). Note that the non-dimensional error index (NDEI) is defined as the ratio of the root mean square error over the standard deviation of the target data and should ideally be 0, while the variance accounted for (VAF) is defined as the ratio between the variance of the real data and the model output and is given out of a maximum of 100 (when the predictions coincide with the real data).

6 Conclusion

In this paper, we have proposed novel strategies and techniques for addressing concept *drift* and *shift* in on-line data streams. To this end, two EFS approaches (*eTS* and *FLEXFIS*) were exploited as on-line modeling methodologies. These were extended by mechanisms which are able to 1.) detect *drifts* and *shifts* with fuzzy rule *ages* and 2.) react to such occurrences appropriately. Reacting applies i) to the rule antecedents by clustering the data space (forcing more significant moves of clusters than in the usual, non-*drift* case); and ii) to rule consequent parameters by including gradual forgetting in the recursive local learning approach. This can be applied to any EFS technique exploiting Takagi-Sugeno type fuzzy systems. Evaluation with real-world data sets showed that the novel techniques are able to improve the accuracy and stability of the fuzzy models, whenever *drifts* and *shifts* occur. Consequently, automatic detection of and reaction to *drifts* and *shifts* may represent an important component in EFS approaches.

Acknowledgements

This work was partially supported by the Upper Austrian Technology and Research Promotion and partially by The Royal Society, UK. This publication reflects only the authors' views.

References

[1] P. Angelov and D. Filev. Simpl_eTS: A simplified method for learning evolving Takagi-Sugeno fuzzy models. In *Proceedings of FUZZ-IEEE 2005*, pages 1068–1073, Reno, Nevada, U.S.A., 2005.

[2] P. Angelov and X. Zhou. Evolving fuzzy-rule-based classifiers from data streams. *IEEE Transactions on Fuzzy Systems, special issue on Evolving Fuzzy Systems*, 16(6):1462–1475, 2008.

[3] P.P. Angelov. Evolving takagi-sugeno fuzzy systems from streaming data, eTS+. In P. Angelov, D. Filev, and N. Kasabov, editors, *Evolving Intelligent Systems: Methodology and Applications*, pp. 21–50. John Wiley & Sons, IEEE Press Series on Computational Intelligence, New York, 2010, ISBN 978-0-470-28719-4.

[4] P.P. Angelov and R.A. Buswell. Evolving rule-based models: A tool for intelligent adaptation. In *9th IFSA World Congress*, pages 1062–1067, 2001.

[5] P.P. Angelov and D. Filev. An approach to online identification of Takagi-Sugeno fuzzy models. *IEEE Trans. on Systems, Man and Cybernetics, part B*, 34(1):484–498, 2004.

[6] P.P Angelov and N. Kasabov. Evolving computational intelligence systems. In *Proceedings of the 1st International Workshop on Genetic Fuzzy Systems*, pages 76–82, Granada, Spain, 2005.

[7] P.P. Angelov and A. Kordon. Adaptive Inferential Sensors based on Evolving Fuzzy Models: An Industrial Case Study. *IEEE Transactions on Systems, Man and Cybernetics, part B: Cybernetics*, 40(2): 529–539, 2010.

[8] P.P. Angelov, E. Lughofer, and X. Zhou. Evolving fuzzy classifiers using different model architectures. *Fuzzy Sets and Systems*, 159(23):3160–3182, 2008.

[9] P.P. Angelov and X.-W. Zhou. Evolving fuzzy systems from data streams in real-time. In *2006 International Symposium on Evolving Fuzzy Systems*, pages 29–35, 2006.

[10] P. Angelov and X.-W. Zhou. On Line Learning Fuzzy Rule-based System Structure from Data Streams. In *2008 IEEE International Conference on Fuzzy Systems* within the *IEEE World Congress on Computational Intelligence*, Hong Kong, June 1-6, 2008, pages 915–922, 2008.

[11] P.P. Angelov *Evolving Rule-based Models: A Tool for Flexible Systems Design*, Springer Physica Verlag, Heidelberg, Germany, February, 2002.

[12] P. Angelov. *Machine Learning (Collaborative Systems)*, patent (WO2008053161, priority date: 1 November 2006; international filing date 23 October 2007; USA publication 11 February 2010, number 2010-0036780).

[13] R. Babuska. *Fuzzy Modeling for Control*. Kluwer Academic Publishers, Boston, 1998.

[14] J. Beringer and E. Hüllermeier. Efficient instance-based learning on data streams. *Intelligent Data Analysis*, 11(6):627–650, 2007.

[15] J. Casillas and O. Cordon and F. Herrera and L. Magdalena. Interpretability Issues in Fuzzy Modeling. *Springer Verlag*, Berlin Heidelberg, 2003.

[16] F. Hopner and F. Klawonn. Obtaining interpretable fuzzy models from fuzzy clustering and fuzzy regression. *Proc. 4th Intern. Conf. on Knowledge-based Intell. Eng. Syst. (KES)*, Brighton, UK, pages pp.162–165, 2000.

[17] S. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2(3):267–278, 1994.

[18] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle. A case-based technique for tracking concept drift in spam filtering. *Knowledge-Based Systems*, 18(4–5):187–195, 2005.

[19] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification - Second Edition*. Wiley-Interscience, Southern Gate, Chichester, West Sussex PO 19 8SQ, England, 2000.

[20] G. Gan and C. Ma and J. Wu. *Data Clustering: Theory, Algorithms, and Applications (Asa-Siam Series on Statistics and Applied Probability)*, Society for Industrial & Applied Mathematics, U.S.A., 2007

[21] R.M. Gray. Vector Quantization. *IEEE ASSP Magazine*, pp. 4–29, 1984.

[22] N. Sundararajan H.-J. Rong, G.-B. Huang, and P. Saratchandran. Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction. *Fuzzy Sets and Systems*, 157(9):1260–1275, 2006.

[23] A. Hensel and T. Spittel. *Kraft- und Arbeitsbedarf bildsamer Formgebungsverfahren*. VEB Deutscher Verlag fr Grundstoffindustrie, 1978.

[24] N. K. Kasabov and Q. Song. DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Trans. on Fuzzy Systems*, 10(2):144–154, 2002.

[25] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, vol. 8 (3), pages 281–300, 2004.

[26] R. Klinkenberg and T. Joachims. Detection concept drift with support vector machines. In *Proc. of the Seventh International Conference on Machine Learning (ICML)*, pages 487–494. Morgan Kaufmann, 2000.

[27] G. Leng, T.M McGinnity, and G. Prasad. An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network. *Fuzzy Sets and Systems*, 150(2):211–243, 2005.

[28] E. Lima, M. Hell, R. Ballini, and F. Gomide. Evolving fuzzy modeling using participatory learning. In P. Angelov, D. Filev, and N. Kasabov, editors, *Evolving Intelligent Systems: Methodology and Applications*, pp. 67–86. John Wiley & Sons, IEEE Press Series on Computational Intelligence, New York, 2010, ISBN 978-0-470-28719-4.

[29] L. Ljung. *System Identification: Theory for the User*. Prentice Hall PTR, Prentic Hall Inc., Upper Saddle River, New Jersey 07458, 1999.

[30] E. Lughofer. Process safety enhancements for data-driven evolving fuzzy models. In *Proceedings of 2nd Symposium on Evolving Fuzzy Systems*, pages 42–48, Lake District, UK, 2006.

[31] E. Lughofer. *Evolving Fuzzy Models — Incremental Learning, Interpretability and Stability Issues, Applications*. VDM Verlag Dr. Müller, Saarbrücken, 2008.

[32] E. Lughofer. Extensions of vector quantization for incremental clustering. *Pattern Recognition*, 41(3):995–1011, 2008.

[33] E. Lughofer. FLEXFIS: A robust incremental learning approach for evolving TS fuzzy models. *IEEE Trans. on Fuzzy Systems (special issue on Evolving Fuzzy Systems)*, 16(6):1393–1410, 2008.

[34] E. Lughofer. Towards robust evolving fuzzy systems. In P. Angelov, D. Filev, and N. Kasabov, editors, *Evolving Intelligent Systems: Methodology and Applications*, pp. 87–126. John Wiley & Sons, IEEE Press Series on Computational Intelligence, New York, 2010, ISBN 978-0-470-28719-4.

[35] E. Lughofer, P. Angelov, and X. Zhou. Evolving single- and multi-model fuzzy classifiers with FLEXFIS-Class. In *Proceedings of FUZZ-IEEE 2007*, pages 363–368, London, UK, 2007.

[36] E. Lughofer and S. Kindermann. Improving the robustness of data-driven fuzzy systems with regularization. In *Proc. of the IEEE World Congress on Computational Intelligence (WCCI) 2008*, pages 703–709, Hongkong, 2008.

[37] E. Lughofer, J. E. Smith, M. A. Tahir, P. Caleb-Solly, C. Eitzinger, D. Sannen, and H. van Brussel. Human-Machine Interaction Issues in Quality Control Based on On-Line Image Classification. In *IEEE Transactions on Systems, Man and Cybernetics, part A: Systems and Humans*, vol. 39(5), pp. 960–971, 2009.

[38] D. Nauck and R. Kruse. NEFCLASS-X – a Soft Computing Tool to Build Readable Fuzzy Classifiers. *BT Technology Journal*, vol. 16 (3): 180–190, 1998

[39] S. Ramamurthy and R. Bhatnagar. Tracking recurrent concept drift in streaming data using ensemble classifiers. In *Proceedings of the Sixth International Conference on Machine Learning and Applications (ICMLA), 2007*, pages 404–409, 2007.

[40] D. Sannen, M. Nuttin, J.E. Smith, M.A. Tahir, E. Lughofer, and C. Eitzinger. An interactive self-adaptive on-line image classification framework. In A. Gasteratos, M. Vincze, and J.K. Tsotsos, editors, *Proceedings of ICVS 2008*, volume 5008 of *LNCS*, pages 173–180. Springer, Santorini Island, Greece, 2008.

[41] J. Sherman and W.J. Morrison. Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix. *Annals of Mathematical Statistics*, 20:621, 1949.

[42] C.F. So, S.C. Ng, and S.H. Leung. Gradient based variable forgetting factor rls algorithm. *Signal Processing*, 83(6):1163–1175, 2003.

[43] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. on Systems, Man and Cybernetics*, 15(1):116–132, 1985.

[44] A. Tsymbal. The problem of concept drift: definitions and related work. Technical Report TCD-CS-2004-15, Department of Computer Science, Trinity College Dublin, Ireland, 2004.

[45] V. Vapnik. *Statistical Learning Theory*. Wiley and Sons, New York, 1998.

[46] L.X. Wang. Fuzzy systems are universal approximators. In *Proc. 1st IEEE Conf. Fuzzy Systems*, pages 1163–1169, San Diego, CA, 1992.

[47] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

[48] S. Wu, M.J. Er, and Y. Gao. A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks. *IEEE Trans. on Fuzzy Systems*, 9(4):578–594, 2001.

[49] R. R. Yager. A model of participatory learning. *IEEE Trans. on Systems, Man and Cybernetics*, 20: 1229–1234, 1990.

[50] R.R. Yager and D.P. Filev. Learning of fuzzy rules by mountain clustering. In *Proc. of SPIE Conf. on Application of Fuzzy Logic Technology*, pages 246–254. Boston, MA, U.S.A., 1993.